Document Overview

The design document shows how the program will run and what each step the program is in as it runs. The UML diagram shows the steps the program will take from Users to printing the pattern to screen and asking the user if he/she wants to save the resulting PNG image or GIF animation. The written document will further explain each step of the program. The class “Function Drawer” is a parameterized function that will take 3 parameters -- a valid function, an image pattern, and time (duration and frequency for animations) -- and will create an image to map the complex function.

Design Descriptions

Users

- User Interface
    - The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals. We expect the users to have a good understanding in the field of mathematics or programming, and therefore, they should need little or no training to use this product.
    - The GUI will be intertwined within a website and feature a working scientific calculator with several drop down features for user input, a sample function list for a user to choose from, and two screens that represent the domain and range when graphing the functions. The user will also be able to save the picture/animation generated.
    - The website will have a menu that will allow the user to access a help section and may have additional extra features such as navigation links.
    - Left side of the GUI:
        - Function field with a drop down arrow. The user can type in their own function or pick from the drop down menu. The drop down menu will contain a list of sample functions. Underneath the function field is the “Enter in Calculator” button. This button brings up the calculator interface, where you can enter your function using buttons instead of typing it. This allows for the user to not worry about getting syntax correct, because the buttons will always use the correct syntax.
        - Pattern field with drop down arrow. The user cannot type a pattern in, but they can choose from the drop down list. Inside the drop down list is a “Custom Image” button, where it will bring up an open file dialog and look for a .PNG file. This file will act as the pattern.
        - Animation field with a checkbox. If checked, a field will appear with required entries in order to perform an animation with the graph. One is a small text entry for the duration of the animation, in seconds. Another is for the frequency, which is how much z is changing per second. Another is the framerate, in frames per second. And the last one is the starting z value.
        - Generate button, that will take all of the parameters given and draw the graph. This button will be grayed out and unclickable

until the parameters are all given. Parameters are: all of the fields mentioned on the left side above as well as a visual position plotted on the domain graph (see below).

- In the middle is the Domain graph. This is a cartesian grid. The user, once a pattern is specified, will place that pattern somewhere on the domain grid. This is a required parameter in order to generate the graph in the Range grid.
- On the right is the Range graph, where the final generated graph is put. Only when the parameters are given, and generate is pushed will the graph show up. The coordinates of the range match the coordinates of the domain.
- Below the range graph is a button “Save as .PNG”, which saves the image to disk. If the range is an animation, the button will change to “Save as .GIF”. In both cases the button will bring up the save dialog box.

Input Function

- Calculator Window Interface
  - The right hand side of the calculator will feature basic calculator functions that are addition, subtraction, multiplication and division. This side will also have a clear function that will erase any input, a backspace function that will delete a single character, an enter key to submit the function to graph, and a memory button. The memory button will save the last 5 functions inputted and will be a drop down feature.
  - The middle of the calculator will feature the exponential, logarithmic functions, and extra basic functions. The exponential features cover the square root, the cubic root, a nth root, a factorial, power of ten, squared function (x^2), cubic function (x^3), and the nth function (x^n). The logarithmic functions include log, the natural log (ln), the e variable (2.718...), and the exp function which is equivalent to e^x. The function Re draws only the real part of whatever is inside the parenthesis (ex. Re(z)). The conjugation function draws the conjugation map of whatever is in the parenthesis (ex. conj(z)).The mod function draws the complex normal of the function drawn in the parenthesis (ex. mod(sin(z))). Finally the abs function draws the absolute value of whatever is in the parenthesis (ex. abs(z*sin(z))
- Sample Functions Dropdown List Interface
  - The user will have the option to choose a sample function instead of writing their own. It will be a drop down list, and will send the pre-set function string directly into the input function class, which will then be sent to the string tokenizer in the same way that a typed function would be sent.

Image Pattern

- *Dropdown List Interface*
  - The user can choose from a pre-set list of patterns in a drop down interface. These are sent to the Image Pattern class, which is then sent as one of the parameters to the function drawer.

- *Browse Pictures Interface*
    - One specific drop down item is the “Browse Pictures” option. This will allow the user to bring up the browse files dialog box. They will choose a .PNG file, and then that will get sent to the Image Pattern. This allows the user to specify their own image pattern and see what their picture looks like when it is warped by their input function.

Animation Parameters

- All of these animation parameters are optional and only go into effect when the user wants an animation.
- *Duration Interface*
    - There will be a small text field where the user can specify how long they want their animation to last before it is looped. The input will be in the form of seconds.
- Frequency Interface
    - This specifies how many units *z* changes per second
- Framerate
    - This specifies how many frames per second
- *Start z value*
    - The *z* value that the animation will start on.

String Tokenizer

- The string tokenizer will read through the input function chosen or typed by the user. The tokens will be read through one by one to check if the function is valid. If the function is not valid, an error message will put on screen for the user. Possible errors could be non matching parenthesis, divide by 0, or missing parenthesis (“sinz” instead of “sin(z)”).
- Data
    - String inputFunction
    - String[] tokens

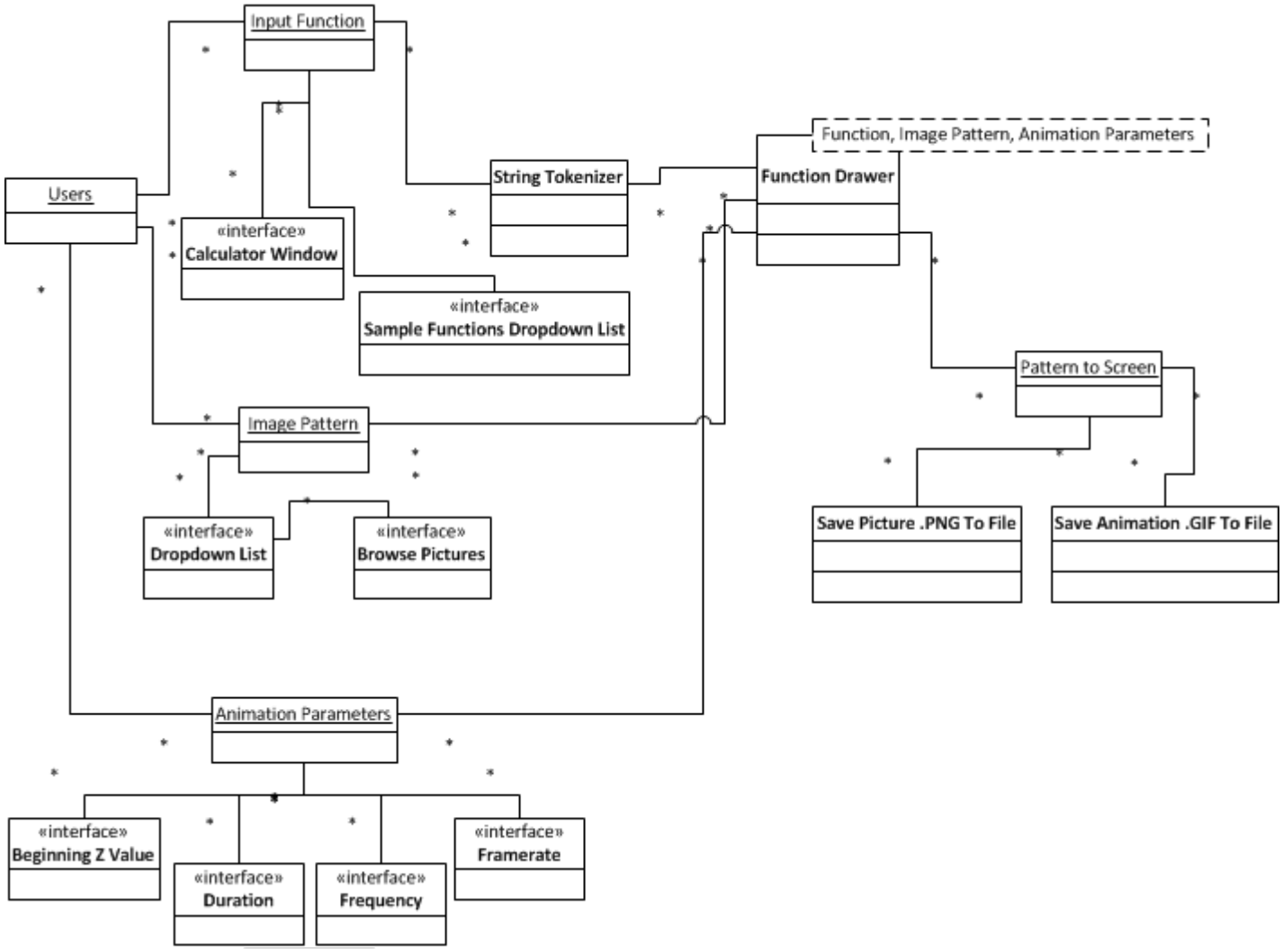
Function Drawer

- The function drawer does the bulk of the mathematics for our program. It takes as parameters the tokenized input function, the image pattern, and an optional time parameter if the user wants animations. When it finishes turning these parameters into a graph image, it sends the result to the Pattern to Screen class, which will do the platform-specific drawing to screen.
- Some steps in the execution of the function drawer based on the knowledge we have now:
    - Calculate the inverse of the given tokenized function (change “z =” to “w =”).
    - Generate a 2D Cartesian grid that matches the coordinates of the pattern.
    - For each pixel on the grid, calculate what color that pixel should be, based on the inverse and on the pattern
    - After going through the grid, send that result to the Pattern to Screen.

Pattern to Screen

- This is the class that will handle all the platform-specific drawing to screen functions. This is in a separate class in preparation for possible implementations on other platforms like HTML5 or flash.
- *Save Picture .PNG to File*
    - A button that appears (or is enabled) after the user generates the function. If the function is not animated, they can save it as a .PNG file. The button will get the image from the Pattern to Screen class, and then bring up the “Save as” dialog.
- *Save Animation .GIF to File*
    - A button that appears(or is enabled) after the user generates the function. If the function IS animated, it can be saved as a .GIF file. The button will get the animation from the Pattern to Screen class, and then bring up the “Save as” dialog.

Program UML Document:



Program Calculator Interface: